Programmation des méthodes Numériques et cours Logiciel Matlab

J.LAASSIRI et S. EL HAJJI

Université Mohammed V - Agdal.
Faculté des Sciences
Département de Mathématiques et Informatique

Laboratoire de Mathématiques, Informatique et Applications
Rabat
http://www.fsr.ac.ma/mia/

Plan :

I) GENERALITES

1) **Qu’est-ce que Matlab ?**

Matlab est un logiciel de calcul scientifique :

- fonctions mathématiques usuelles
- calcul matriciel
- racines d’un polynôme
- équations différentielles
- intégration numérique
- graphiques 2D & 3D
- etc.

Il peut être doté de nombreuses extensions (boîtes à outils : statistique, SIMULINK, …).
http://www.mathworks.com

2) On peut utiliser Matlab de deux manières différentes

a) en entrant des instructions à la suite du prompteur >>, le logiciel se comporte comme une très bonne calculatrice graphique ;

b) en créant des **scripts** (ou **m-files**) ou des fonctions (extension **.m**), on crée des “ programmes ” sauvegardables.

3) Aide

Pour tout problème de syntaxe utilisez l’aide en ligne (commande help).

II) OPERATIONS NUMERIQUES EN LIGNE

Il s'agit ici d'utiliser les opérateurs mathématiques afin de se familiariser avec le logiciel. Le calcul sur les nombres complexes est aussi abordé.

1) Premières commandes

Voici quelques exemples à essayer (chercher le résultat avant l'exécution de la ligne et le commenter) :

>> **3*4**

On peut taper plusieurs commandes Matlab sur une même ligne, en les séparant par une virgule :

>> **5*6, 2^5**
>> **3+5*2^5**

Les opérateurs arithmétiques ont la priorité habituelle (cf. calculette).

>> **3+5*2^5/5**

Pour rappeler des commandes précédentes, 2 possibilités :
Utiliser les touches du clavier ↑ ou ↓ et les commandes précédentes apparaissent dans l'ordre chronologique ;
Taper les premiers caractères de la commande puis utiliser les touches ↑ ou ↓.
Après avoir choisi la commande à rappeler, il est toujours possible de la modifier en se déplaçant avec les touches ← et → et en effectuant la correction

>> **(3+5*2^5)/5**

Une ligne d'instruction terminée par un point virgule est exécutée immédiatement mais le résultat n'est pas affiché :

>> **(3+5*2^5)/5 ;**

Exemples d'assignation de variables (inutile de les prédéclarer)

>> **x=2**

Il faut rappeler le nom de la variable pour voir s'afficher les résultats :

>> **x**
>> **y=x^5**
>> **y/x**

Il est important de bien comprendre le sens du signe = qui ne signifie pas qu'il y a identité entre les deux termes. Il signifie que le résultat de l'expression de droite est assigné à la variable de gauche qui prend donc une nouvelle valeur.
A titre d'exemple, x=x+1 n'a de sens qu'en langage de programmation : le résultat de la somme du “ contenu ” de la variable x et de 1 est recopié dans la même variable x.

La dernière réponse est appelée **ans**, à défaut de lui avoir donné un nom. On peut l'utiliser ainsi (deviner les résultats) :

>> **z=3*ans, ans, z=4*z**

2) Calculs sur les nombre complexes

Dans Matlab, i (ou j) désigne le nombre imaginaire pur de partie imaginaire égale à 1 ; si la variable i a été utilisée entre temps à un autre usage (indice de boucle par exemple) vous pouvez la réinitialiser par i=sqrt(-1) .

>> **i^2**
>> **j^2**
>> **z1=1+sqrt(3)*i** (sqrt : racine carrée)
>> **z1c=conj(z1)** (donne le conjugué de z1)

>> **theta=angle(z1)** (donne l'argument de z1 en radian)
>> **theta=theta*180/pi** (convertir en degré, pi=3.14…)
>> **r = abs(z1)** (donne le module de z1)
>> **realZ1=real(z1)**
>> **imagZ1=imag(z1)**
Pour les autres opérations sur les complexes voir l'aide en ligne.

3) Affichage des résultats

Essayez :
>> a=sqrt(3)
>> format long, b=sqrt(3)
>> a-b
>> format short

A tout moment, vous pouvez consulter la liste actuelle des variables de votre espace de travail :

```
>> who
>> whos
```

La commande `whos` *fournit en plus des informations sur la taille, le nombre d'éléments et le nombre d'octets occupés. Matlab travaille toujours sur des données en* ***double précision*** *(****soit 8 octets****).*

4) Matrices

Le "Mat" de Matlab ne signifie pas Mathématique mais Matrice. De fait, il considère tout nombre réel comme une matrice 1×1. Pour Matlab, tout est tableau.

Syntaxe : A = [a_{11}, a_{12} ; a_{21}, a_{22}], avec

$$\begin{matrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{matrix}$$

- Les éléments d'une ligne sont séparés par des blancs ou des virgules,
- Les éléments d'une colonne sont séparés par un point virgule ou un retour chariot,
- Le tout est mis entre crochets.

L'appel aux éléments de la matrice est le suivant : **A(lignes, colonnes)**

a) Création

>> **a=[1,2,3;4,5,6]** % tableau à deux dimensions, 2 lignes×3 colonnes

b) Manipulation

>> **a(1,2), a(2,3)** % deux éléments du tableau, repérés par leurs indices ligne et colonne
>> **a(2,3)=10** % le nombre 10 est assigné à un élément du tableau
>> **a'** % matrice adjointe
>> **c=[a;7,8,9]**
>> **a=[1:6;2:7;4:9]**
>> **a, a(1,:), a(:,2)**

c) Destruction des variables précédentes

>> **clear all**
vérifier que les variables sont bien détruites.

d) Vecteur (= matrice ou tableau à une dimension)

L'opérateur ":" est très utile :

>> **-3:3**
>> **x=0:10**
>> **x=-3:.3:3**
>> **x(2:12)**
>> **x(5)**
>> **length(x)** (dimension du vecteur)

e) Calculs

Il faut noter que l'on peut multiplier, diviser et faire toute sorte d'opérations sur les tableaux en utilisant les opérateurs usuels. La très grande majorité des fonctions mathématiques peuvent opérer avec des tableaux. Si les tableaux ne sont pas de même taille, quelques précautions sont à prendre

Si l'on veut diviser chaque élément du tableau a par l'élément correspondant du tableau b,

```
>>a=[2 4 6];
>>b=[1 2 3];
>>a./b
ans =

   2    2    2
```

alors que

```
>>a/b
ans =    2
```

Il est possible de réaliser des opérations terme à terme sur les matrices (les matrices sont alors assimilées à des tableaux).

Les deux opérateurs + et – travaillent déjà terme à terme. Pour les autres opérateurs *, /, \, et ^ , il est nécessaire de les précéder d'un point pour préciser que les opérations se font terme à terme.

Remarque :

*Notez quelques symboles importants : **pi**, **i** (pour un nombre complexe) et **eps** (2.2e-16, précision maximale des calculs). Il faut également noter que ces symboles peuvent voir leur contenu modifié. Leur sens donné plus haut est celui fixé au lancement de Matlab. Si, après une modification maladroite, on veut revenir à la valeur prédéfinie, il faut taper **clear** nom de la variable.*

III) FONCTIONS GRAPHIQUES

Matlab offre la possibilités de résultats graphiques assez esthétiques. Travaillant sous Windows, ils peuvent être recopiés dans une autre application (Word, ...)

1) Fonctions 2D

\>> **x=-10:.1:10;**
\>> **plot(x.^2)** % chaque élément du tableau x à la puissance 2 et non produit matriciel
\>> **figure** % créer une nouvelle fenêtre graphique. permet de stocker plusieurs graphes en une même session
\>> **plot(x, x.^2)**

✓ **Les styles de tracé :**

Symbole	Couleur
y	yellow
m	magenta
c	cyan
r	red
g	green
b	blue
w	white
k	black

Symbole	Style
-	trait plein
:	pointillé
-.	tiret/pointillé
--	tiret
none	pas de ligne

Symbole	Marqueur
+	
o	cercle
*	astérisque
.	point
x	croix
square	carré
diamond	carreau
^	triangle pointe en haut
v	triangle pointe en bas
<	triangle pointe à gauche
>	triangle pointe à droite
pentagram	étoile à 5 pointes
hexagram	étoile à 6 pointes
none	pas de marqueur

Exemple :
\>> **plot(x, x.^2, ‘+’)**
\>> **plot(x, x.^2, ‘c*’)**

Pour voir les figures : menu *Windows.*

\>> **xlabel('x')**
\>> **ylabel('y=x^2')**
\>> **figure**
\>> **plot(x.^2, x)**
\>> **plot(x, x.*sin(x))**
\>> **figure**
\>> **plot(x.*cos(x),x.*sin(x))** % exemple de courbe paramétrée
\>> **figure**
\>> **comet(x.*cos(x),x.*sin(x))**

2) Fonctions 3D

On utilise la commande plot3. Elle possède la même syntaxe que plot avec le paramètre z d’altitude en plus :
\>> **t = 0:pi/50:20*pi;**
\>> **plot3(sin(t),cos(t),t);**

IV) EXERCICES

1) COURBE DE GAUSS

Tracer la courbe correspondant à la fonction $y = \exp\left(-\frac{(x-5)^2}{2}\right)$ pour $0 \le x \le 10$.

(on commencera par créer un tableau de valeur pour x avec un pas de 0.01).

2) DOSAGE COLORIMETRIQUE DU MANGANESES DANS UN ACIER

Objectif :

- Tracer la courbe d'absorbance **A=f(λ)** à partir de résultats expérimentaux
- Exploiter cette courbe pour obtenir des résultats par interpolation

λ (nm)	**A**
400	0.073
410	0.053
420	0.046
430	0.048
440	0.059
450	0.075
460	0.104
470	0.159
480	0.216
490	0.313
500	0.397
510	0.507
520	0.569
530	0.638
540	0.581
550	0.606
560	0.408
570	0.358
580	0.224

a) Créer les tableaux **lambda** et **A** ;
b) Tracer A=f(lambda) (afficher les point expérimentaux uniquement) ;
c) Tracer la courbe par interpolation avec un pas de 1nm. Pour cela, utiliser la commande **spline** (voir l'aide : ***help spline***) pour l'interpolation et **hold** pour afficher les 2 courbes sur le même graphique;
d) Déterminer A pour λ quelconque (à l'aide de **spline**).
e) Trouver le maximum (fonction **max**).

V) AIDE POUR TP

1) LE SCRIPT AIDE_TP1

```
%Le script aide_TP1 en mode "démonstration" permet de se
%familiariser avec les commandes simples de Matlab.
%On fait défiler la suite du contenu de cette aide en appuyant
%sur n'importe quelle touche.
%Les commentaires écrits après un "%" ne sont pas interprétés.
%Les premières lignes de commentaires sont disponible par "help aide_TP1"
help aide_TP1   %on affiche les premières lignes de commentaires ci-dessus
echo on         %mode démonstration avec affichage des instructions
pause
format compact  %affichage des résultats sans sauts de ligne

disp(' '), disp(' '), disp('Variables :'), disp(' ')
a=1
2+a
b=3*ans             %le dernier résultat est appelé "ans" pour "answer"
1 + 2^5/4, 3e6,...   %"," sépare sur une ligne et "..." empêche la rupture de ligne
i^2, pi, exp(i*pi)   %i et j représentent (sauf si redéfinis) la même racine de -1
a=1; b=a+1; c='0*a'; %";" permet de ne pas afficher le résultat:
a=a, b, 2*c          %la chaîne de 3 caractères "c" est une matrice 1x3
x=(1>=0); y=(0~=0);
x, y                %vrai=1 et  faux=0
pause

disp(' '), disp(' '), disp('Affichage :'), disp(' ')
format long, 1/3          %format avec (au maximum) 15 chiffres
format short e, 1/3       %format avec 5 chiffres en notation scientifique
format, 1/3               %format courant (5 décimales, non compact)
format compact
1+eps==1, 1+eps/2==1      %epsilon machine = eps = 2^(-52)
2^(-1074)                 %plus petit nombre realmin flottant > 0
2^(1023)*(2-2^(-52))      %plus grand nombre realmax flottant
10^1000, ans/ans          %infini : Inf, not_a_number : NaN
pause

disp(' '), disp(' '), disp('Matrices et vecteurs :'), disp(' ')
A=[1 2;2 3]     %cette matrice s'écrit aussi [1,2;2,3]
A(2,2)=1;  A
B=rand(2,3)     %matrice 2x3 aléatoire
u=2:4           %":" permet de construire des suites arithmétiques
2:0.5:4         %nombres de 2 a 4 par pas de 0.5
v=[2 3 4]'      %vecteur = adjoint d'une matrice ligne
```

```
prod(1:20)      %factorielle 20
p=poly([0,-2])  %coefficients en décroissant du polynôme p de racines 0 et -2
roots(p)        %récupération des racines de p dans C
a0=p(length(p))  %coefficient "constant" de p
z=polyval(p,2);  %évaluation de p(2)
fprintf('\np(2)=%g\n\n',z) %affichage du résultat avec saut de ligne par "\n"
pause

disp(' '), disp(' '), disp('Tests et boucles :'), disp(' ')
clear x; for k=1:5 x(k)=k^2; end; x'  %les vecteurs 1:n sont d'accès très rapide
k=0; while k^3<10; k=k+1; end; k       %utiliser plutôt si possible "for"
n = input('nombre entier à tester ?\n'); %dialogue
if rem(n,2)==0, r='pair'; else  r='impair'; end;
fprintf('\n%g est %s\n\n',n,r)
pause

disp(' '), disp(' '), disp('Graphes :'), disp(' ')
figure(1), clf          %on prépare la fenêtre 1 (inutile si pas "hold on")
x=[-.5:0.001:0.5]';     %représentation de l'axe des abscisses par une suite de
points
y=x.*sin(x);
plot(x,y,'r'), axis equal %graphe de x->x*sin(1/x) sur [-1/2,1/2] en rouge ("red")
title('y=x*sin(x)'), xlabel('x'), ylabel('y'), grid   %légende et grille
v=axis; line([v(1),v(2)],[0,0]) %on trace l'axe des abscisses
pause

disp(' '), disp(' '), disp('Équations différentielles :'), disp(' ')
figure(2), clf          %on prépare la fenêtre 2
[t1,Y1] = ode45('secondmembre_y',[0 1],[1]);  %y' = y sur [0,1] avec y(0)=1
[t2,Y2] = ode45('secondmembre_y',[0 -1],[1]); %y' = y sur [-1,0] avec y(0)=1
plot([t2 t1],[Y2 Y1]); %on trace la solution de y' = y sur [-1,1] avec y(0)=1
title('solution de y''=y et y(0)=1')

clear all   %on efface de la mémoire les variables affectées
echo off
```

1) LE SCRIPT AIDE_TP2

```
%Le script aide_TP2 permet de se familiariser avec
%les commandes simples de Matlab concernant les matrices.
help aide_TP2, format short, format compact, echo on
pause
```

```
disp(' '), disp(' '), disp('Matrice adjointe, produit :'), disp(' ')
clear i, u=[0 i 2]     %variante : "u=[0,i,2]"
u', u.'               %matrice adjointe et transposée
3*u, u/2, u'*u, u*u'
pause

disp(' '), disp(' '), disp('Id, diagonale, zeros et uns :'), disp(' ')
A=[1 2; 3 4]         %variante : 1ère ligne avec "A=[1 2" et 2ème avec "3 4]"
eye(size(A)), eye(2,3) %la matrice identité "I" se lit, comme "eye" en anglais
diag(A), diag([1 4])  %extraction de diagonale et construction de matrice
diagonale
zeros(2,3), ones(3,2)
pause

disp(' '), disp(' '), disp('Taille, norme, parties réelle et imaginaire :'), disp(' ')
u=[0 i 2], size(u,1), size(u,2), length(u) %"1"="ligne", "2"="colonne"
norm(u,1), norm(u,2), norm(u,'inf'), norm(u)
A=[1 2; 3 4], size(A), size(A,1), size(A,2)
norm(A,1), norm(A,2), norm(A,'inf'), norm(A,'fro')
z=1+2*i
norm(z), abs(z), real(z), imag(z), conj(z)
pause

disp(' '), disp(' '), disp('Déterminant, valeurs propres :'), disp(' ')
A=[1 2; 3 4], det(A)
norm(A,2), sqrt(max(eig(A'*A)))
norm(A,2)==sqrt(max(eig(A'*A))) %il ne faut pas croire le résultat de ce test
p=poly([1,1]), C=compan(p)     %une matrice de pol. car. proportionnel à p
eig(C), poly(C)               %valeurs propres et polynôme caractéristique

disp(' '), disp(' '), disp('Matrice par blocs :'), disp(' ')
A=[1 2; 3 4]
[A,[7 8]'], [A;[7 8]]   %variantes: "[A [7 8]']" et "[A;7 8]"
pause

disp(' '), disp(' '), disp('Puissance :'), disp(' ')
A=[1 2; 3 4]
inv(A), eye(size(A))/A  %cf. "help slash"
B=A^2
B^0.5, ans^2           %une racine carrée de B autre que A
pause
```

```
disp(' '), disp(' '), disp('Opérations terme à terme :'), disp(' ')
A=[1 2; 3 4]
A+1
A.^2
sqrt(A)      %matrice A.^0.5 des racines carrées des coefficients
exp(A)       %par contre, l'exponentielle de la matrice A est "expm(A)"
A.*A
A./A
1 ./A       %à ne pas confondre avec l'inverse de A
pause

disp(' '), disp(' '), disp('Extraction de sous-matrices :'), disp(' ')
A=[1 2; 3 4]
A(1,2), A(1,1), A(1,[2,1])  %des coefficients de A
A(1,:), A(:), A(:,[2,1])    %":" donne tous les éléments
A(1,end), A(:,end-1)      %"end" donne le dernier élément
pause

disp(' '), disp(' '), disp('Matrices creuses :'), disp(' ')
S=sparse([1 2 3],[ 2 4 1],[1,2,3]) %indices i, indices j, puis valeurs
T=full(S), sparse(T)
U=T(1:2,2:4)
D1=diag(ones(2,1),-1), D2=diag(ones(2,1),1)

clear all
echo off
```

```
%Le script aide_TP2 permet de se familiariser avec
%les commandes simples de Matlab concernant les matrices.
help aide_TP2, format short, format compact, echo on
pause

disp(' '), disp(' '), disp('Matrice adjointe, produit :'), disp(' ')
clear i, u=[0 i 2]     %variante : "u=[0,i,2]"
u', u.'             %matrice adjointe et transposée
3*u, u/2, u'*u, u*u'
pause
```

```
disp(' '), disp(' '), disp('Id, diagonale, zeros et uns :'), disp(' ')
A=[1 2; 3 4]          %variante : 1ère ligne avec "A=[1 2" et 2ème avec "3 4]"
eye(size(A)), eye(2,3) %la matrice identité "I" se lit, comme "eye" en anglais
diag(A), diag([1 4])  %extraction de diagonale et construction de matrice
diagonale
zeros(2,3), ones(3,2)
pause

disp(' '), disp(' '), disp('Taille, norme, parties réelle et imaginaire :'), disp(' ')
u=[0 i 2], size(u,1), size(u,2), length(u) %"1"="ligne", "2"="colonne"
norm(u,1), norm(u,2), norm(u,'inf'), norm(u)
A=[1 2; 3 4], size(A), size(A,1), size(A,2)
norm(A,1), norm(A,2), norm(A,'inf'), norm(A,'fro')
z=1+2*i
norm(z), abs(z), real(z), imag(z), conj(z)
pause

disp(' '), disp(' '), disp('Déterminant, valeurs propres :'), disp(' ')
A=[1 2; 3 4], det(A)
norm(A,2), sqrt(max(eig(A'*A)))
norm(A,2)==sqrt(max(eig(A'*A))) %il ne faut pas croire le résultat de ce test
p=poly([1,1]), C=compan(p)      %une matrice de pol. car. proportionnel à p
eig(C), poly(C)                 %valeurs propres et polynôme caractéristique

disp(' '), disp(' '), disp('Matrice par blocs :'), disp(' ')
A=[1 2; 3 4]
[A,[7 8]'], [A;[7 8]]   %variantes: "[A [7 8]']" et "[A;7 8]"
pause

disp(' '), disp(' '), disp('Puissance :'), disp(' ')
A=[1 2; 3 4]
inv(A), eye(size(A))/A  %cf. "help slash"
B=A^2
B^0.5, ans^2            %une racine carrée de B autre que A
disp(' '), disp(' '), disp('Opérations terme à terme :'), disp(' ')
A=[1 2; 3 4]
A+1
A.^2
sqrt(A)       %matrice A.^0.5 des racines carrées des coefficients
exp(A)        %par contre, l'exponentielle de la matrice A est "expm(A)"
A.*A
```

```
A./A
1 ./A          %à ne pas confondre avec l'inverse de A
disp(' '), disp(' '), disp('Extraction de sous-matrices :'), disp(' ')
A=[1 2; 3 4]
A(1,2), A(1,1), A(1,[2,1])  %des coefficients de A
A(1,:), A(:), A(:,[2,1])    %":" donne tous les éléments
A(1,end), A(:,end-1)        %"end" donne le dernier élément
disp(' '), disp(' '), disp('Matrices creuses :'), disp(' ')
S=sparse([1 2 3],[ 2 4 1],[1,2,3]) %indices i, indices j, puis valeurs
T=full(S), sparse(T)
U=T(1:2,2:4)
D1=diag(ones(2,1),-1), D2=diag(ones(2,1),1)

clear all
echo off
```

PROGRAMMATION

Le but de cette partie est d'utiliser Matlab comme un langage de programmation, en se familiarisant avec les instructions de contrôle insérées dans un **script** ou une **fonction**.

Avec Matlab, les commentaires sont précédés du signe ' % ' ; ce qui veut dire que ceux-ci ne sont pas pris en compte par Matlab.

1) Les instructions d'affectation

- Exemple 1 :

  y=9

  x=y+10 *% il y a affectation du résultat de y+10 à x*

  % il y a un sens : ce qui est à droite du signe = est placé dans la variable de gauche

- Exemple 2 :

  x=sqrt(2) *% il y a calcul de* $\sqrt{2}$ *puis affectation du résultat à x*

La même variable peut apparaître des deux côtés du signe =

- Exemple :

  x=x*10 *% il y a d'abord calcul de x*10 puis affectation du résultat à x qui change ainsi de valeur*

Dans l'instruction où il y a affectation, il est nécessaire que ce qui est à gauche du signe = soit une variable et non une expression de variable.

Le signe = n'est pas un signe d'identité, mais un signe d'affectation.

- Exemple :

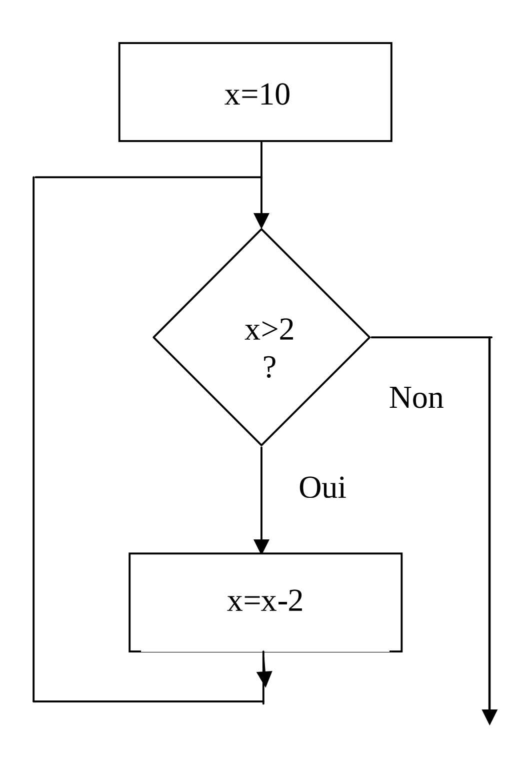
  x+2=y+10 *% n'a pas de sens, x+2 n'est pas une variable*

2) Les instructions de contrôle

Les instructions de contrôle sont à la base de tout calcul numérique. Il est donc indispensable de les bien comprendre.

a) L'instruction « while »

```
while expression est vraie,
        instructions;
end
```



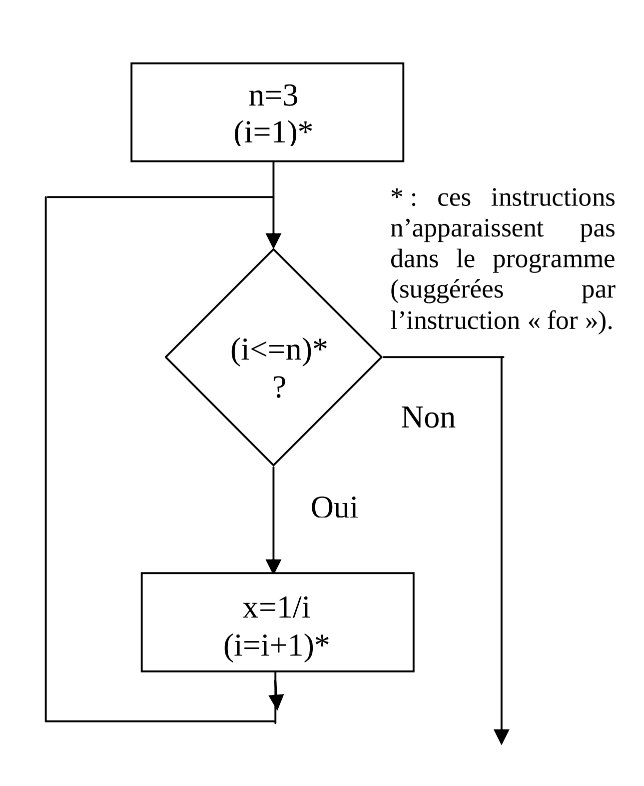
Le test est fait avant l'instruction (noter la virgule après l'expression logique).
Il s'agit d'une instruction de boucle : tant que l'expression n'est pas fausse les instructions sont exécutées **en boucle**.
Il est évident que, pour pouvoir sortir de cette boucle, les instructions doivent pouvoir modifier la valeur de l'expression.

<u>exemple</u> :

x=10;
while x>2, *% tant que (x>2) est vrai, c'est à dire x>2*
x=x-2; *% x est décrémenté de 2*
end
x *% affiche x*

b) L'instruction "for"

```
for expression,
        instructions;
end
```



C'est une instruction de boucle pour laquelle l'expression adopte une forme particulière (noter la virgule à la fin de la ligne « for »).
Contrairement à l'instruction **while**, le nombre de boucles est fixé avant, par « expression ».

<u>exemple</u> :

n=3;
for i=1:n, *% pour i variant de 1 à 3 par pas de 1*
x=1/i;
end
x

c) L'instruction if

```
if expression1 est vraie
        instructions1;
else    instructions3
end
```

S'il y a plus de 2 possibilités :

```
if expression1 est vraie
        instructions1;
elseif expression2 est vraie
        instructions2;
else    instructions3
end
```

C'est un test et non une instruction de boucle.

<u>exemple</u> :

Affecter préalablement un entier aux variable i et x

```
if i<3
        x=x+1;
else    x=x-3;
end
 x
```

3) **Exemple de script**

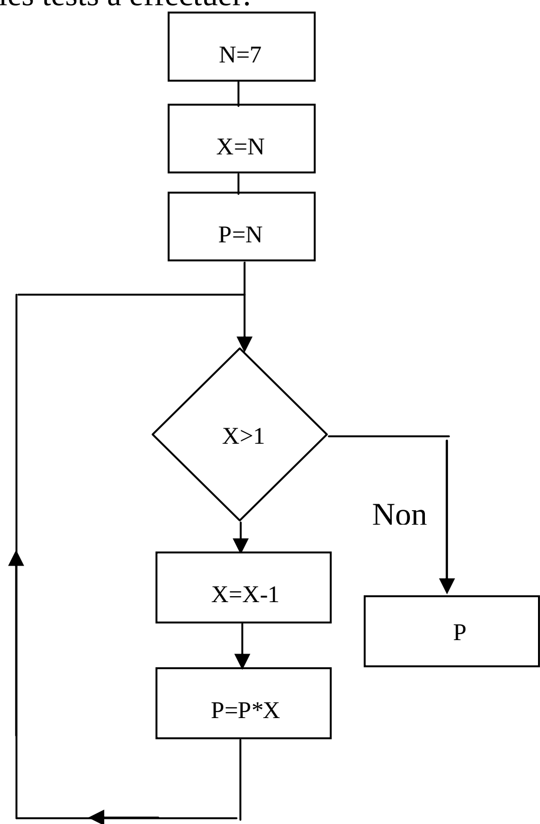
On choisit comme illustration le calcul de la factorielle d'un nombre entier positif non nul N (N=7 par exemple).
Rappel : N ! = N.(N-1).(N-2)……2.1

Principe : La méthode consiste à partir de N et à le décrémenter jusqu'à 1. Tant que l'on a pas atteint cette dernière valeur, on multiplie la valeur obtenue au produit des précédentes.

1ère étape : On commence par évaluer les variables nécessaires :

- N,
- la valeur après décrémentation X,
- et le produit des valeurs précédentes P.

2nde étape : Organigramme :
La factorielle est la valeur finale de P. On crée un organigramme qui représente les instructions, les boucles et les tests à effectuer.



3ème étape : Ecriture du script dans le bloc-note (File-New …). Sauvegarder dans le dossier de votre groupe (exemple C:\A1) avec l'extension « **.m** » par exemple **ex0.m** (m-file). Attention, par défaut, l'éditeur de texte sauvegarde avec l'extension « **.txt** ».

```
N=7 ;
X=N ;
P=N ;
while X>1,
        X=X-1;
        P=P*X;
end
P
```

4ème étape : Exécution. Avant d'exécuter il faut spécifier le répertoire où se trouve le m-file, en écrivant l'instruction suivante dans la fenêtre matlab : **cd c:\A1**. Dans la fenêtre de commande Matlab, taper le nom du fichier sans l'extension; par exemple **ex0**.

4) Exemple de fonction

A la différence du fichier de script, la fonction Matlab prend des paramètres d'entrée et retourne des paramètres de sortie. ***Le fichier doit porter le même nom que la fonction***.

Exemple : définition de la fonction f(x)= $\frac{\sin^2(x)}{x^2}$

Lancer l'éditeur de texte de Matlab puis taper les lignes suivantes :

```
function y=f(x)
y=(sin(x)).^2./(x.^2) ;
```

Sauvegarder ensuite ce fichier sous le nom **f.m** puis exécutez-le :

```
» x=-10:0.1:10;
» y=f(x);
» plot(x,y) ;
```

5) Exercices

Pour les exercices suivants, on passera par les mêmes étapes.

a) Exercice 1

Écrire un programme calculant cos(x) par l'intermédiaire de la série :

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

Le calcul itératif s'arrêtera quand le dernier terme à additionner sera inférieur ou égal en valeur absolue (**abs()**) à 10^{-5}.

Chaque terme de la somme sera évalué en fonction du terme précédent : $u_n = -\frac{x^2 \cdot u_{n-2}}{n(n-1)}$ pour n pair $n \geq 2$ et $u_0 = 1$. On ne calcule pas les factorielles.

b) Exercice 2 : recherche de zéro par dichotomie

Dans l'intervalle [0,2], la fonction $f(x) = x^5 + 3 \cdot x^2 - 10$ est croissante et de signe contraire aux bornes. On cherche la valeur de la fonction au milieu de l'intervalle (ici 1), et suivant le signe du résultat ($f(1) < 0$), on travaille dans l'un ou l'autre des demi-intervalles (ici [1,2]). On itère ce calcul ($f(1.5) > 0$ jusqu'à l'obtention d'une précision suffisante.

c) Exercice 3 : Calcul d'intégrale par la méthode des trapèzes

Calcul de l'intégrale d'une fonction f(x) par la méthode des trapèzes.
La fonction sera le polynôme $6x^2+2x+1$ dont l'intégrale entre 0 et 1 vaut 4.
On appelle a et b les bornes de l'intervalle.
On divisera l'intervalle en n=1000 pas. On utilise comme variable x les différentes valeurs x_i possibles entre a et b par pas de $h = \frac{b-a}{n}$: $x_1 = a$ et $x_n = b$

On montre que, par cette méthode, on peut estimer l'intégrale par :

$$\int_a^b f(x)dx \approx h\left[\sum_1^n \frac{\left(f(x_i)+f(x_{i+1})\right)}{2}\right] = h\left[\sum_1^{n+1} f(x_i) - \frac{\left(f(a)+f(b)\right)}{2}\right]$$

d) Exercice 4 : Monte-Carlo

On estime la valeur de π par la méthode de Monte-Carlo : on lance des points (de coordonnées x et y) au hasard dans un carré de côté égal à 2 qui contient un cercle de diamètre égal à 2. Les points pouvant tomber avec la même probabilité partout dans le carré, le rapport du nombre de points dans le cercle sur le nombre de points total tend vers le rapport des surfaces :

$$\frac{\text{Nombre de points dans le cercle}}{\text{Nombre total de points}} \rightarrow \frac{\text{Surface du cercle}}{\text{Surface du carré}} = \frac{\pi \cdot r^2}{d^2} = \frac{\pi}{4}$$

Pour un nombre donné N de points (par exemple 10000) tirés au hasard, on cherchera le nombre de ces points contenus dans le cercle, c'est-à-dire tels que leur module est inférieur au rayon R :

$$\sqrt{x^2+y^2} \leq R$$

Pour simplifier les calculs, on pourra s'intéresser à un quart de cercle, les valeurs de x et de y étant alors positives.

L'instruction $x = rand$ ***renvoie un nombre réel positif pseudo-aléatoire inférieur à1.***

e) Exercice 5 : suite de Fibonacci

La suite de Fibonacci u_n est décrite par l'équation récurrente suivante :

$$u_{n+2}=u_{n+1}+u_n \text{ avec } u_0=0 \text{ et } u_1=1$$

On montre que $\lim_{n\to\infty}\left[\frac{u_{n+1}}{u_n}\right] = \Phi = \frac{1+\sqrt{5}}{2} = 1.618$ (« nombre d'or »).
Estimez ce nombre par un calcul numérique avec une précision de 10^{-3}.

6) **<u>Série des Exercices :</u>**

a) **<u>Série</u>**

Exercice 1 :

On note u1=[1 ,2, 3], u2=[-5, 2 ,1], u3=[-1, -3, 7]

- **Définir ces vecteurs sous Matlab.**
- **Calculer u1+ u2, u1 + 3u2 - 5u3, u3/5**
- **Calculer norm1(u1); norm2(ul); norm1(u2); norminf(u3);**
- **Calculer le cosinus de l'angle formé par les vecteurs U1et U2.**

Exercice 2 :

Calculer les déterminants, inverses, valeurs propres et vecteurs propres de chacune des matrices

A=[2 3 ; 6 5] et B=[2 3 4 ;7 6 5 ;2 8 7]

Exercice 3 On note

$$A = \begin{pmatrix} 5/8 & -1/4 & 1/8 \\ 1/4 & 0 & 1/4 \\ 1/8 & -1/4 & 5/8 \end{pmatrix}, b = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}, u_0 = \begin{pmatrix} 5 \\ 2 \\ -4 \end{pmatrix}$$

et on définit, pour $n \geq 0$, la suite de vecteurs $u_{n+1} = Au_n + b$.

1. Calculer les premiers termes de la suite u_n, qu'observez-vous?

2. Même question avec et Interpréter les résultats

$$B = \begin{pmatrix} 5 & 6 & 3 \\ -1 & 5 & -1 \\ 1 & 2 & 0 \end{pmatrix}, b = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}, u_0 = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}$$

Exercice 4

1. Soit la matrice A symétrique définie par:

$$A = \begin{pmatrix} 1.4025 & 0.5975 & -1.3404 & 1.0921 \\ * & 0.538 & 1.2141 & -0.774 \\ * & * & 0.0096 & 2.907 \\ * & * & * & 0.0499 \end{pmatrix}$$

   (a) Calculer le déterminant, la trace et l'inverse de la matrice A.

   (b) Calculer le polynôme caractéristique de A, chercher ses racines.

   (c) Rechercher les valeurs propres et les vecteurs propres de A. Vérifier sur un de ces vecteurs qu'il est bien vecteur propre.

   (d) Calculer le conditionnement de A.

2. Soit $x = (1111)$. Calculer sa norme 1, sa norme 2, sa norme ∞.

Exercice 5 On considère la matrice tridiagonale d'ordre n définie par

$$A_n = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}$$

- Que fait la séquence d'instructions suivante ?

1	2
$S = [eye(n)\ zeros(n,1)];$	$D = diag(ones(n,1));$
$S = S(:, 2:n+1);$	$SD = diag(ones(n-1,1),1);$
$A = 2*eye(n) - S - S'$	$A = 2*D - SD - SD';$

1. **construire** ***An*** par concaténation et extraction !
 Pour plus d'informations, taper help **eye**, help **zeros** dans la ligne de commande.
2. **construire*An*** en utilisant la commande **diag**,
Pour plus d'informations, taper help **diag** dans la ligne de commande.
3. Il *est* recommandé de sauvegarder dans un fichier une suite d'instructions que l'on veut utiliser plusieurs fois ; ainsi la procédure "**proc**" est un ensemble d'instructions contenues le fichier de nom "**proc.m**", et qui seront exécutées en tapant simplement proc dans la ligne de commande de **Matlab.**
4. Ecrire une procédure pour construire la matrice A pour n quelconque et résoudre ensuite un système linéaire $\boldsymbol{Ax = b}$.
5. Mesurer le temps calcul pour plusieurs valeurs de n, à l'aide de la commande **cputime** (ou encore **tic** et **toc**).
6. Faire l'expérience pour une matrice A définie en stockage plein (tableau à deux dimensions), puis en stockage creux **(A=sparse(n,n)).**

7. Déterminer de manière expérimentale le nombre d'opérations $N(n)$ que nécessite une résolution de système en fonction de n, pour le stockage plein et le stockage creux. Tracer la courbe $N(n)$ en échelle naturelle (utiliser la commande plot), puis en échelle log-log (utiliser la commande loglog).
   • A l'aide de Matlab, proposer une méthode expérimentale permettant de montrer que la **matrice** ***A*** **est définie positive**.
   • **Représenter graphiquement en échelle log-log** le conditionnement de la matrice A en fonction de n. Commenter.

b) Serie2

Objectif On veut résoudre des Systèmes linéaires.

Méthodes directes - Méthodes itératives.

1. Ecrivez un programme pour résoudre le système d'équations suivant

$x_1 + 2x_2 + 3x_3 + 4x_4 = 1$
$2x_1 - 3x_2 + 4x_3 + 5x_4 = 2$
$3x_1 + 4x_2 - 5x_3 + 6x_4 = 3$
$4x_1 + 5x_2 + 6x_3 - 7x_4 = 4$

Par la méthode de décomposition **U^tDU** de **Choleski** de sa matrice. La matrice du système est-elle définie positive? Comparez votre résultat à celui que fournit La procédure standard de Matlab.

2. Sans écrire un programme général résolvez le système suivant

$9x_1 - 2x_2 - 2x_3 = 11$
$2x_1 + 2x_2 + 8x_3 = 13$
$2x_1 - 9x_2 + 2x_3 = 15$

a) par la méthode de **Jacobi**,
b) par la méthode de **Gauss Seidel**
Comparez les résultats obtenus du point de vue de la vitesse de convergence. Concluez.

3. Les méthodes itératives de Jacobi et Gauss-Seidel s'écrivent respectivement

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^{n} a_{ij} x_j^{(k)} \right), \qquad i = 1, \dots, n,$$

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \qquad i = 1, \dots, n.$$

Ecrivez un programme qui permet de résoudre un système linéaire par la méthode de **Jacobi** et résolvez le système suivant à l'aide de ce programme:

$9x_1 - 2x_2 - 2x_3 + 2x_4 + 2x_5 = 1$
$2x_1 + 0x_2 + 8x_3 - 2x_4 + 2x_5 = 2$
$2x_1 - 9x_2 + 2x_3 + 2x_4 - 2x_5 = 3$
$0x_1 - 2x_2 + 2x_3 + 2x_4 + 8x_5 = 4$
$2x_1 + 2x_2 + 2x_3 - 9x_4 - 2x_5 = 5$

Modifiez votre programme pour appliquer la méthode de **Gauss- Seidel** au système précédent. Comparez les résultats obtenus.

c) Serie3 :

Objectif On veut résoudre Equations non linéaires.

Exercice 1:

a)donner l'algorithme de Méthode de Newton pour résoudre f(x)=0

b) Ecrire la fonction Newton.m : function x=Newton(x0, n)

% initialisation a x0, on fait n itérations.

%la fonction f et sa différentielle df sont respectivement définies dans les fichiers fnewton.m et df.m.

c) Ecrire la fonction convergence_newton.m :

% pour le polynôme $\mathbf{P=x^4+x^3-23x^2+3x+90}$, étudier la vitesse de convergence de la méthode de Newton % a partir de x0=0, on tombe sur la racine -5.

Exercice 2 : L'équation : **x + 2.cos(x)=0** possède trois racines réelles.
Localisez ces racines au moyen d'un graphe.
Calculez une valeur approchée de ces racines avec au moins huit chiffres précis, par la méthode d'itération du point fixe et par la méthode de Newton.
Comparez les résultats obtenus et estimez la vitesse de convergence dans chaque cas. Que concluez-vous?

Exercice 3 :.L'équation $\mathbf{f(x) = e^X-x^2/2-x-1}$
possède une racine multiple dans l'intervalle **(-0.5, 1).**
Calculez une valeur approchée de cette racine par application de la méthode
a) de **dichotomie**,
b) itérative du **point fixe**,
c) de la **sécante**,
d) de **Newton**.
Commentez vos résultats.